

**APPARATUS AND METHOD FOR STORING DATA INTO INCOMPATIBLY
FORMATTED STORAGE SYSTEMS**

5

BACKGROUND OF THE INVENTION

1. Technical Field:

10 The present invention is directed to a method and apparatus for managing data storages. More specifically, the present invention is directed to a method and apparatus for storing a piece of data into incompatibly formatted storage systems.

2. Description of Related Art:

15 In recent years, computer systems have experienced great advances in processor speed, performance, available memory, and system architecture. Often times, they are rendered obsolete within months of being announced. Large customers, such as, major corporations and other
20 institutions that want or need to keep their systems at or near state-of-the-art performance levels are constantly upgrading their computer systems. Generally, these large customers tend to maintain brand loyalty. This often occurs for a variety of reasons of which the following are but a
25 few: (1) no need to retrain personnel, (2) data compatibility, (3) cost. The present invention provides a solution to the data compatibility issue and, thus, affords these customers a greater degree of freedom to choose among a plurality of computer systems when upgrading to newer and
30 faster systems.

One of the reasons for data incompatibility is that some computer systems format data differently than others.

In addition, this differently formatted data is often stored into storage systems that are themselves formatted differently than those used by the other computer systems. For example, Sun Microsystems computer systems running Solaris operating system (OS) format and store data differently than IBM computer systems running AIX OS, although both Solaris and AIX OSs are UNIX-based OSs. Hence, data from a Sun Microsystems computer system is not compatible with an IBM computer system and vice versa.

Consequently, when a customer upgrades from one set of computer systems to another set that is incompatible with the first set, the older machines are not ordinarily used anymore. The data that was generated by the older computer systems may continue to be used if converted into a format compatible with the new computer systems. However, as is well known in the field, most conversion algorithms are not ideal and therefore, often times not all the data is converted.

What is needed, therefore, is an apparatus and method of storing copies of data into incompatibly formatted data storage systems, thus, allowing customers to upgrade to newer computer systems and have previously-stored data be available without conversion.

SUMMARY OF THE INVENTION

The present invention provides a method, system and apparatus for storing data into two incompatibly formatted storage systems. When a data is to be stored, it is first determined whether the data is to be stored into the two incompatibly formatted storage systems, which are each managed by a logical volume manager (LVM). If so, the data is forwarded to the two storage systems to be stored. In a particular embodiment, when the data is to be stored, one of the two LVMs determines whether the data is to be stored into the two storage systems. If so, a copy of the data will be transferred to the other LVM for storage. The two LVMs then ensure that the data is appropriately stored.

When data is to be read from an incompatible storage system, the first LVM may request the data from the second LVM, when the data is received, the first LVM converts the data into an understandable format before the data is used.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Fig. 1 is an exemplary block diagram illustrating a distributed data processing system according to the present invention.

Fig. 2 is an exemplary block diagram of a server apparatus according to the present invention.

Fig. 3 is an exemplary block diagram of a client apparatus according to the present invention.

Fig. 4 is a conceptual view of an LVM.

Fig. 5 depicts two storage management systems for two computer systems.

Fig. 6 is a flow chart of a process that may be used to enable mirrored copies of a piece of data to be stored in two incompatible physical storage systems.

Fig. 7 is a flow diagram of a process that may be used to retrieve data from an incompatible storage system. The process starts when data is to be read.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

With reference now to the figures, Fig. 1 depicts a pictorial representation of a network of data processing systems in which the present invention may be implemented. Network data processing system 100 is a network of computers in which the present invention may be implemented. Network data processing system 100 contains a network 102, which is the medium used to provide communications links between various devices and computers connected together within network data processing system 100. Network 102 may include connections, such as wire, wireless communication links, or fiber optic cables.

In the depicted example, server 104 is connected to network 102 along with storage unit 106. In addition, clients 108, 110, and 112 are connected to network 102. These clients 108, 110, and 112 may be, for example, personal computers or network computers. In the depicted example, server 104 provides data, such as boot files, operating system images, and applications to clients 108, 110 and 112. Clients 108, 110 and 112 are clients to server 104. Network data processing system 100 may include additional servers, clients, and other devices not shown. In the depicted example, network data processing system 100 is the Internet with network 102 representing a worldwide collection of networks and gateways that use the TCP/IP suite of protocols to communicate with one another. At the heart of the Internet is a backbone of high-speed data communication lines between major nodes or host

computers, consisting of thousands of commercial, government, educational and other computer systems that route data and messages. Of course, network data processing system 100 also may be implemented as a number of different
5 types of networks, such as for example, an intranet, a local area network (LAN), or a wide area network (WAN). Fig. 1 is intended as an example, and not as an architectural limitation for the present invention.

Referring to Fig. 2, a block diagram of a data
10 processing system that may be implemented as a server, such as server 104 in Fig. 1, is depicted in accordance with a preferred embodiment of the present invention. Data processing system 200 may be a symmetric multiprocessor (SMP) system including a plurality of processors 202 and 204
15 connected to system bus 206. Alternatively, a single processor system may be employed. Also connected to system bus 206 is memory controller/cache 208, which provides an interface to local memory 209. I/O bus bridge 210 is connected to system bus 206 and provides an interface to I/O
20 bus 212. Memory controller/cache 208 and I/O bus bridge 210 may be integrated as depicted.

Peripheral component interconnect (PCI) bus bridge 214 connected to I/O bus 212 provides an interface to PCI local bus 216. A number of modems may be connected to PCI local
25 bus 216. Typical PCI bus implementations will support four PCI expansion slots or add-in connectors. Communications links to network computers 108, 110 and 112 in Fig. 1 may be provided through modem 218 and network adapter 220 connected to PCI local bus 216 through add-in boards.

Additional PCI bus bridges 222 and 224 provide interfaces for additional PCI local buses 226 and 228, from which additional modems or network adapters may be supported. In this manner, data processing system 200 allows connections
5 to multiple network computers. A memory-mapped graphics adapter 230 and hard disk 232 may also be connected to I/O bus 212 as depicted, either directly or indirectly.

Those of ordinary skill in the art will appreciate that the hardware depicted in Fig. 2 may vary. For example,
10 other peripheral devices, such as optical disk drives and the like, also may be used in addition to or in place of the hardware depicted. The depicted example is not meant to imply architectural limitations with respect to the present invention.

15 The data processing system depicted in Fig. 2 may be, for example, an IBM e-Server pSeries system, a product of International Business Machines Corporation in Armonk, New York, running the Advanced Interactive Executive (AIX) operating system or LINUX operating system.

20 With reference now to Fig. 3, a block diagram illustrating a data processing system is depicted in which the present invention may be implemented. Data processing system 300 is an example of a client computer. Data processing system 300 employs a peripheral component
25 interconnect (PCI) local bus architecture. Although the depicted example employs a PCI bus, other bus architectures such as Accelerated Graphics Port (AGP) and Industry Standard Architecture (ISA) may be used. Processor 302 and main memory 304 are connected to PCI local bus 306 through
30 PCI bridge 308. PCI bridge 308 also may include an integrated memory controller and cache memory for processor 302. Additional connections to PCI local bus 306 may be

made through direct component interconnection or through add-in boards. In the depicted example, local area network (LAN) adapter 310, SCSI host bus adapter 312, and expansion bus interface 314 are connected to PCI local bus 306 by direct component connection. In contrast, audio adapter 316, graphics adapter 318, and audio/video adapter 319 are connected to PCI local bus 306 by add-in boards inserted into expansion slots. Expansion bus interface 314 provides a connection for a keyboard and mouse adapter 320, modem 322, and additional memory 324. Small computer system interface (SCSI) host bus adapter 312 provides a connection for hard disk drive 326, tape drive 328, and CD-ROM drive 330. Typical PCI local bus implementations will support three or four PCI expansion slots or add-in connectors.

An operating system runs on processor 302 and is used to coordinate and provide control of various components within data processing system 300 in Fig. 3. The operating system may be a commercially available operating system, such as Windows 2000, which is available from Microsoft Corporation. An object oriented programming system such as Java may run in conjunction with the operating system and provide calls to the operating system from Java programs or applications executing on data processing system 300. "Java" is a trademark of Sun Microsystems, Inc. Instructions for the operating system, the object-oriented operating system, and applications or programs are located on storage devices, such as hard disk drive 326, and may be loaded into main memory 304 for execution by processor 302.

Those of ordinary skill in the art will appreciate that the hardware in Fig. 3 may vary depending on the implementation. Other internal hardware or peripheral devices, such as flash ROM (or equivalent nonvolatile

memory) or optical disk drives and the like, may be used in addition to or in place of the hardware depicted in Fig. 3. Also, the processes of the present invention may be applied to a multiprocessor data processing system.

5 As another example, data processing system 300 may be a stand-alone system configured to be bootable without relying on some type of network communication interface, whether or not data processing system 300 comprises some type of network communication interface. As a further example, data
10 processing system 300 may be a Personal Digital Assistant (PDA) device, which is configured with ROM and/or flash ROM in order to provide non-volatile memory for storing operating system files and/or user-generated data.

The depicted example in Fig. 3 and above-described
15 examples are not meant to imply architectural limitations. For example, data processing system 300 may also be a notebook computer or hand held computer in addition to taking the form of a PDA. Data processing system 300 also may be a kiosk or a Web appliance.

20 The present invention provides an apparatus and method of storing data into incompatibly formatted storage systems, thus, allowing sets of incompatible computer systems to use the data. Although the invention may preferably be local to server 104, it may nonetheless, be local to client systems
25 108, 110 and 112 of Fig. 1 or to both the server 104 and clients 108, 110 and 112. Consequently, the present invention may reside on any data storage medium (i.e., floppy disk, compact disk, hard disk, ROM, RAM, etc.) used by a computer system.

30 Most servers use a plurality of physical storage systems. To fully take advantage of these physical storage systems, the servers make use of an abstraction layer called

a logical volume manager (LVM) when storing and retrieving data. Volume, in this case, is the storage capacity of a physical storage system. Thus, volume and physical storage system will henceforth be used interchangeably.

5 The LVM arranges the physical storage systems into volume groups in order to give the impression that storage systems having each a much more voluminous storage capacity are being used. Within each volume group, one or more logical volumes may be defined. Data stored in a logical
10 volume appears to be stored contiguously. In actuality the data may be interspersed throughout many different locations across all the physical storage systems that make up the volume group.

 Stated differently, each logical volume in a logical
15 volume group is divided into logical partitions. Likewise, each physical volume in a volume group is divided into physical partitions. Each logical partition corresponds to at least one physical partition. But, although the logical partitions in a logical volume are numbered consecutively or
20 appear to be contiguous to each other, the physical partitions to which they each correspond need not be contiguous to each other. And indeed, most often, the physical partitions are not contiguous to each other. Thus, one of the many tasks of the LVM is to keep tab on which
25 physical partition or partitions correspond to a logical partition.

 For fault tolerance and performance, some servers store more than one copy of each piece of data onto the physical storage systems they use. Storing two or more copies of a
30 piece of data is called mirroring the data. In order to mirror data, each logical partition must correspond to as many physical partitions as there are mirrors (or copies) of

the data. In other words, if the data is to be mirrored three times, each logical partition has to correspond to three physical partitions.

Fig. 4 is a conceptual view of an LVM. In Fig. 4 three layers are depicted, an application layer 400, a logical layer 410 and a physical layer 420 each having one or more devices. It should be noted that the devices shown in the three layers are not all inclusive. There may be more devices in use in each of the application layer 412, the logical layer 410 and the physical layer 430. Thus, the devices in Fig. 4 should be taken only as an example of devices that may be used.

The logical layer 410, for all intent and purpose, is the LVM. The LVM may be regarded as being made up of a set of operating system commands, library subroutines or other tools that allow a user to establish and control logical volume storage. The LVM controls physical storage system resources by mapping data between a simple and flexible logical view of the storage space and the actual physical storage system. The LVM does this by using a layer of device driver code that runs above traditional device drivers. This logical view of the disk storage is provided to application programs and is independent of the underlying physical disk structure.

The logical layer 410 contains a logical volume 412 that interacts with logical volume device driver 414. A device driver, as is well known in the art, acts as a translator between a device and programs that use the device. That is, the device driver accepts generic commands from programs and translates them into specialized commands for the device. In this case, the logical volume device driver 414 translates commands from an application program

executing on the computer system for device driver 430. When an application program sends commands to file system manager 402 to store or retrieve data from logical volume 412, the file system manager 402 informs the logical volume
5 manager 412 of the application program's wish. The logical volume manager 412 then conveys the wish to the logical volume device driver 414. The logical volume device driver 414 then consults the appropriate mirroring map, policies etc. and instructs the device driver 430 which ones of
10 physical storage systems 422, 424, 426 and 428 to use for the data.

Thus, if the data being stored is to be mirrored, the data will be stored according to an appropriate mirroring map (e.g., data from each logical partition will be stored
15 in more than one physical partition of the storage systems as per the mirroring map). The mirroring map is usually entered into the LVM by a system administrator. Mirroring data is described in APPARATUS AND METHOD OF CREATING A MIRRORING MAP FOR USE WITH DIFFERENT PIECES OF DATA, serial
20 number _____, filed on _____ (IBM Docket AUS920010862US1), the disclosure of which is herein incorporated by reference.

In any case, if the data is being read, only one mirrored copy will be used. If for any reason the mirrored copy cannot be read, another mirrored copy will instead be
25 read. The present invention stores mirrored data into two incompatibly formatted physical storage systems. To do so, two LVMs are used.

Fig. 5 depicts two storage management systems that are used on a computer system. The two storage management
30 systems are dissimilar and incompatibly formatted. Logical volume device driver 506 of LVM 510 is connected to logical volume device driver 556 of LVM 560. Accordingly, when a

piece of data being stored on the computer system is to be mirrored and if the data is to be stored into the two physical storage systems, the logical volume device driver 506 will forward a copy of the data to logical volume device driver 556. The logical volume device driver 556 will then instruct device driver 558 to store the data into any or all of physical storage systems 562, 564, 566 and 568 as per a storage map. Meanwhile the other mirrored copy or copies of the data will be stored into any or all of physical storage systems 512, 514, 516 and 518 of the first computer system.

Fig. 6 is a flow chart of a process that may be used to enable mirrored copies of a piece of data to be stored in two incompatible physical storage systems. The process starts whenever data is to be stored (step 600). Then, LVM₁, by consulting its map, determines whether the data is to be mirrored. If so, LVM₁ again determines whether one copy of the data is to be stored in the other physical storage system. If so, the copy is forwarded to LVM₂ (steps 600, 605, 615 and 620). When LVM₂ receives the data, it will instruct its own device driver to store the data accordingly.

LVM₁ will also instruct its device driver to store the data in accordance with its mirroring map. If the data is not to be mirrored or if the data is to be mirrored only within the first physical storage system, LVM₁ then merely needs to instruct its device driver to store the data as per the map (step 610).

Fig. 7 is a flow diagram of a process that may be used to retrieve data from an incompatible storage system. The process starts when data is to be read. Then LVM₁ makes a check to determine whether the data is stored in its own physical storage system. If so, it merely has to instruct

its device driver to retrieve the data. The retrieved data can then be forwarded to the File system manager (steps 705 and 710). If the data resides in the other physical storage system, LVM₁ will request that LVM₂ convey the data. At 5 that instant LVM₂ will instruct its own device driver to retrieve the data. Once retrieved, the data will be forwarded to LVM₁. When LVM₁ receives the data, it will have to convert the data in a format that is understandable to the computer system. When done, the data will then be 10 forwarded to the file system manager (steps 705, 715, 720 and 725).

The description of the present invention has been presented for purposes of illustration and description, and is not intended to be exhaustive or limited to the invention 15 in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. The embodiment was chosen and described in order to best explain the principles of the invention, the practical application, and to enable others of ordinary skill in the art to 20 understand the invention for various embodiments with various modifications as are suited to the particular use contemplated.